Exercise: 6

Name: Pekka Lehtola

How many tasks did you do: 8

Were the tasks easy, ok, difficult: Easy

Do you need help/comments in any task (if yes, to which ones):

1. Explain the following terms:

a. Super class

Super luokalla tarkoitetaan sitä luokkaa mistä ollaan peritty, käytetään myös nimitystä parent

b. Sub class

Sub class tarkoittaa luokkaa mikä on peritty Super classista.

c. Base class

Oman ymmärrykseni mukaan, Base class = Super class

d. Derived class

Derived class = Sub class

e. "Is a" relationship

Tällä tarkoitetaan suhdetta luokkien välillä.

Dog **is a** domestic_animal. Domestic_animal **is a** Mammal

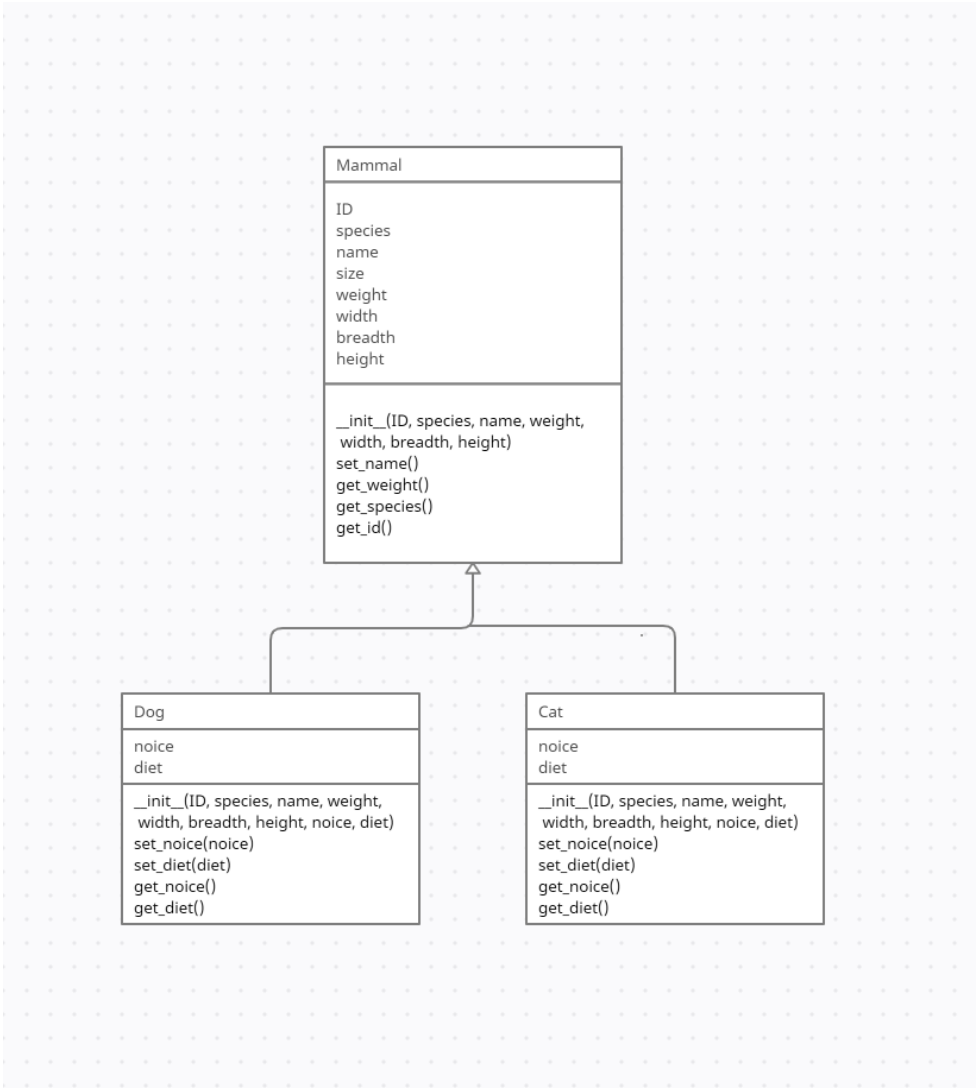
Platypus **is a** Wild_animal. Wild_animal **is a** Mammal

Student **is a** participant of OOP course.

Teacher **is a** participant of OOP course.

Jos tämä ei käy järkeen luokissa on jonkinlainen logiikka virhe.

2. Draw a UML diagram of task 3. Use UML syntax and see how data attributes and methods are presented in the example presented on class (about Automobile), can be found on lecture slides. You can use any software or e.g. draw by hand and take a picture. Example in lectures is drawn using MS Visio.



3. Inherit some animals from the Mammal class (that you created in Exercise 4). Add data attribute for the noise the animal makes and the diet they have. Display your objects on screen (= Print out the state of each object (use str-method)).

Screen capture of Task 3

Mammal class:

main.py | mammal_class.py

```
# File name: mammal_class
# Author: Pekka Lehtola
# Description: class for creating mammals

class Mammal:

    def __init__(self):

        self.id = "ID"
        self.species = "species"
        self.name = "name"
        self.weight = "weight"
        self.width = 1
        self.breadth = 1
        self.height = 1
        self.size = float((self.width * self.breadth * self.height) / (1000 * 1000)) #Calculates cm^3 and converts it to m^3

    #str method for clean output printing with correct units
    def __str__(self):

        return f"""
    ID: {self.id}
    Species: {self.species}
    Name: {self.name}
    Size {self.size} m^3
    Weight {self.weight} Kg
    Width {self.width} cm
    Breadth {self.breadth} cm
    Height {self.height} cm
        """

    def set_id(self):
        self.id = int(input("Enter an id for the animal: "))
    def set_species(self):
        self.species = (input("Enter a species for the animal: "))
    def set_name(self):
        self.name = str(input("Enter a name for the animal: "))
    def set_weight(self):
        self.weight = int(input("Enter a weight for the animal: "))
    def set_width(self):
        self.width = int(input("Enter a width for the animal: "))
        self.size = float((self.width * self.breadth * self.height) / (1000 * 1000))
    def set_breadth(self):
        self.breadth = int(input("Enter a breadth for the animal: "))
        self.size = float((self.width * self.breadth * self.height) / (1000 * 1000))
    def set_height(self):
        self.height = int(input("Enter a height for the animal: "))
        self.size = float((self.width * self.breadth * self.height) / (1000 * 1000))

    def get_id(self):
        return self.id
    def get_species(self):
        return self.species
    def get_name(self):
        return self.name
    def get_weight(self):
        return self.weight
    def get_width(self):
        return self.width
    def get_breadth(self):
        return self.breadth
    def get_height(self):
        return self.height
```

Main:

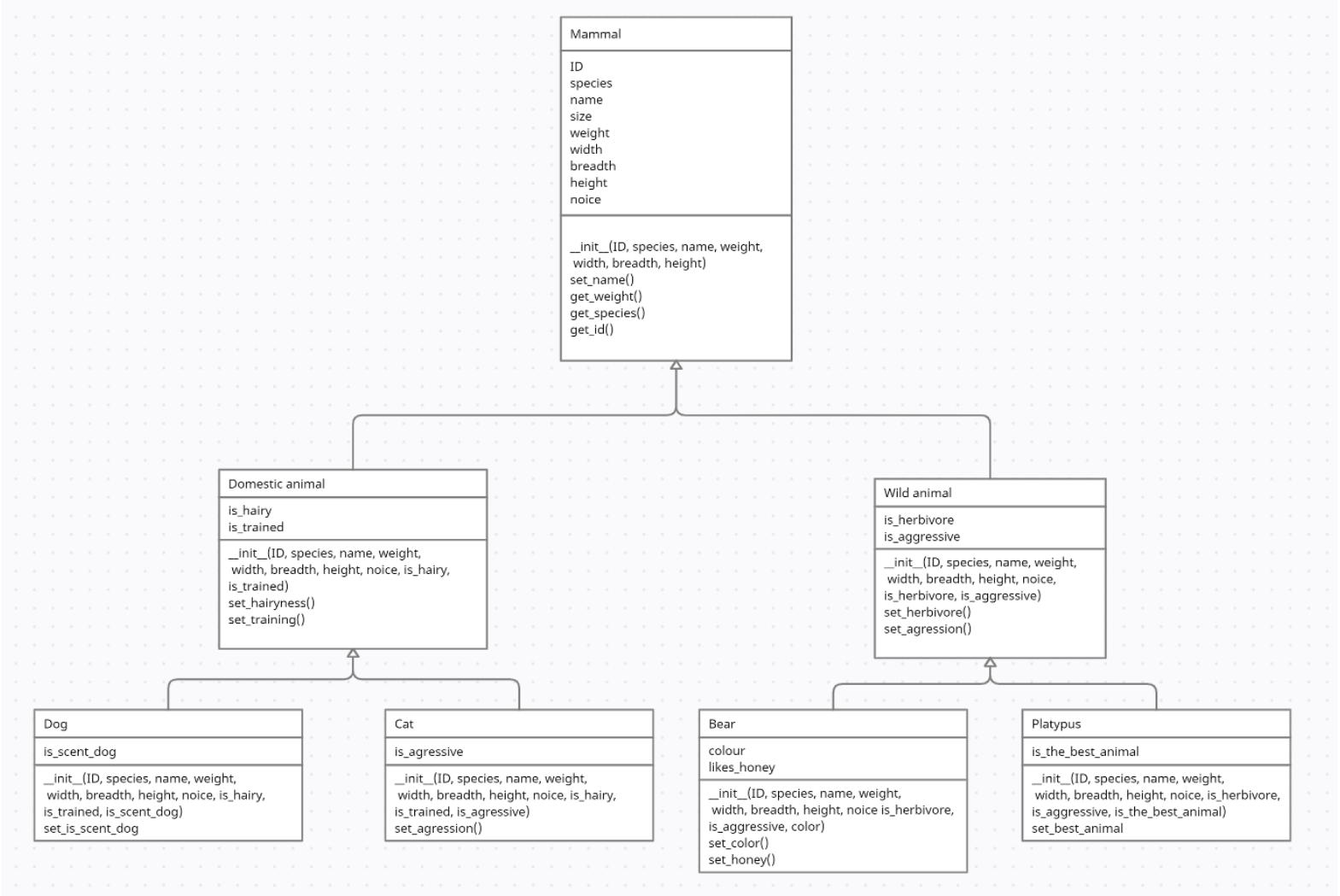
main.py mammal_class.py

```python
# File name: main
# Author: Pekka Lehtola
# Description: Main function for exercise 6_3

from mammal_class import Mammal

#Dog inherited from Mammal class
class Dog(Mammal):

    #Two new attributes noice and diet
    #Rest are inherited
    def __init__(self):
        Mammal.__init__(self)
        self.noice = "noice"
        self.diet = "diet"

    #Set and Get methods
    def set_noice(self):
        self.noice = str(input("Enter a noice for the animal: "))
    def set_diet(self):
        self.diet = str(input("Enter a diet for the animal: "))

    def get_noice(self):
        return self.noice
    def get_diet(self):
        return self.diet

    #Str function for clear output prints.
    def __str__(self):

        return f"""
    ID: {self.id}
    Species: {self.species}
    Name: {self.name}
    Size: {self.size} m^3
    Weight: {self.weight} Kg
    Width: {self.width} cm
    Breadth: {self.breadth} cm
    Height: {self.height} cm
    Noice: {self.noice}
    Diet is {self.diet}
        """

#Setting dog as object and giving it values.
dog = Dog()

dog.set_id()
dog.set_species()
dog.set_name()
dog.set_weight()
dog.set_width()
dog.set_breadth()
dog.set_height()
dog.set_diet()
dog.set_noice()

print(dog)
```

Screen capture of the output of Task 3

```
C:\Users\pekka\AppData\Local\Microsoft\WindowsApps\python3.7.exe "C:/Users/pekka/OneDrive -
Enter an id for the animal: 1
Enter a species for the animal: Dog
Enter a name for the animal: Mauri
Enter a weight for the animal: 66
Enter a width for the animal: 32
Enter a breadth for the animal: 50
Enter a height for the animal: 50
Enter a diet for the animal: 300g dog food twice a day
Enter a noice for the animal: Bark

    ID: 1
    Species: Dog
    Name: Mauri
    Size: 0.08 m^3
    Weight: 66 Kg
    Width: 32 cm
    Breadth: 50 cm
    Height: 50 cm
    Noice: Bark
    Diet is 300g dog food twice a day


Process finished with exit code 0
```

4. Draw a UML diagram of exercise 5. Use UML syntax and see how data attributes and methods are presented in the example presented on class (about Automobile etc.).



5. Change your task 3 code like this: Inherit a domestic animal from Mammal. Also inherit a wild animal from Mammal. Then inherit a few domestic and wild animals from those classes and print them out. Each mammal should make unique noise and have a certain diet as additional data attributes. Add some relevant attributes. Display your objects on screen.

Screen capture of Task 5

Mammal_class:

```
# File name: mammal_class
# Author: Pekka Lehtola
# Description: class for creating mammals

class Mammal:

    def __init__(self):

        self.id = "ID"
        self.species = "species"
        self.name = "name"
        self.weight = "weight"
        self.width = 1
        self.breadth = 1
        self.height = 1
        self.size = float((self.width * self.breadth * self.height) / (1000 * 1000)) #Calculates cm^3 and converts it to m^3
        self.noice = ""
        self.diet = ""

    #str method for clean output printing with correct units
    def __str__(self):

        return f"""
    ID: {self.id}
    Species: {self.species}
    Name: {self.name}
    Size: {self.size} m^3
    Weight: {self.weight} Kg
    Width: {self.width} cm
    Breadth: {self.breadth} cm
    Height: {self.height} cm
    Noice: {self.noice}
    Diet is {self.diet}
        """

    def set_id(self, id):
        self.id = int(id)
    def set_species(self, species):
        self.species = species
    def set_name(self, name):
        self.name = name
    def set_weight(self, weight):
        self.weight = int(weight)
    def set_width(self, width):
        self.width = int(width)
        self.size = float((self.width * self.breadth * self.height) / (1000 * 1000))
    def set_breadth(self, breadth):
        self.breadth = int(breadth)
        self.size = float((self.width * self.breadth * self.height) / (1000 * 1000))
    def set_height(self, height):
        self.height = int(height)
        self.size = float((self.width * self.breadth * self.height) / (1000 * 1000))
    def set_noice(self, noice):
        self.noice = noice
    def set_diet(self, diet):
        self.diet = diet

    def get_id(self):
        return self.id
    def get_species(self):
        return self.species
    def get_name(self):
        return self.name
    def get_weight(self):
        return self.weight
    def get_width(self):
        return self.width
    def get_breadth(self):
        return self.breadth
    def get_height(self):
        return self.height
    def get_noice(self):
        return self.noice
    def get_diet(self):
        return self.diet
```

Animals:

main.py | domestic_animals.py | wild_animals.py | animals.py | mammal_class.py

```python
# File name: animals
# Author: Pekka Lehtola
# Description: Classes Wild_animal and Domestic_animal Inherited from Mammal class.

from mammal_class import *

#Inherits Wild_animals from Mammal class.
class Wild_animal(Mammal):

    #Wild animals have additional attributes Herbivore and Agression.
    def __init__(self):
        Mammal.__init__(self)
        self.is_herbivore = True
        self.is_agressive = False

    def set_herbivore(self):

        if self.is_herbivore:
            self.is_herbivore = False
        else:
            self.is_herbivore = True

    def set_agression(self):

        if self.is_agressive:
            self.is_agressive = False
        else:
            self.is_agressive = True

# Inherits Domestic_animal from Mammal class.
class Domestic_animal(Mammal):

    #Domestic animals have additional attributes Hairyness and Training.
    def __init__(self):
        Mammal.__init__(self)
        self.is_hairy = True
        self.is_trained = True

    def set_hairyness(self):

        if self.is_hairy:
            self.is_hairy = False
        else:
            self.is_hairy = True

    def set_training(self):

        if self.is_trained:
            self.is_trained = False
        else:
            self.is_trained = True
```

Wild_animals:

main.py | domestic_animals.py | wild_animals.py | animals.py | mammal_class.py

```python
# File name: wild_animals.
# Author: Pekka Lehtola
# Description: Bear and Platypus classes inherited from Wild_animals class.

from animals import *

# Inherits Bear from Wild_animals class
class Bear(Wild_animal):

    def __init__(self):

        Wild_animal.__init__(self)
        self.colour = "Brown"
        self.likes_honey = True

    #Additional Attribute colour for bear class
    def set_colour(self, colour):
        self.colour = colour

    # Calling this function changes the boolean value.
    def set_likes_honey(self):

        if self.likes_honey:
            self.likes_honey = False
        else:
            self.likes_honey = True

    #Bear class printing
    def __str__(self):

        return f"""
    ID: {self.id}
    Species: {self.species}
    Name: {self.name}
    Size: {self.size} m^3
    Weight: {self.weight} Kg
    Width: {self.width} cm
    Breadth: {self.breadth} cm
    Height: {self.height} cm
    Noice: {self.noice}
    Diet is {self.diet}
    is herbivore: {self.is_herbivore}
    is agressive: {self.is_agressive}
    Colour is: {self.colour}
    Likes honey: {self.likes_honey}
        """

#Inherits Platypus from Wild_animal class
class Platypus(Wild_animal):

    def __init__(self):

        Wild_animal.__init__(self)
        self.is_the_best_animal = True

    # Calling this function does nothing because platypus is the Superior animal.
    def set_is_the_best_animal(self):

        if self.is_the_best_animal:
            self.is_the_best_animal = True
        else:
            self.is_the_best_animal = True

    #Printing for platypus class.
    def __str__(self):

        return f"""
    ID: {self.id}
    Species: {self.species}
    Name: {self.name}
    Size: {self.size} m^3
    Weight: {self.weight} Kg
    Width: {self.width} cm
    Breadth: {self.breadth} cm
    Height: {self.height} cm
    Noice: {self.noice}
    Diet is {self.diet}
    is herbivore: {self.is_herbivore}
    is agressive: {self.is_agressive}
    Is the best animal: {self.is_the_best_animal}
        """
```

Domestic_animals:

main.py | domestic_animals.py | wild_animals.py | animals.py | mammal_class.py

```python
# File name: domestic_animals
#Author: Pekka Lehtola
# Description: Dog and Cat classes inherited from Domestic_animals class.

from animals import *

# Inherits Dog from Domestic animals.
class Dog(Domestic_animal):

    def __init__(self):

        Domestic_animal.__init__(self)
        self.is_scent_dog = True

    # Calling this function changes the boolean value.
    def set_is_scent(self):

        if self.is_scent_dog:
            self.is_scent_dog = False
        else:
            self.is_scent_dog = True

    # Defining Dog printing.
    def __str__(self):

        return f"""
    ID: {self.id}
    Species: {self.species}
    Name: {self.name}
    Size: {self.size} m^3
    Weight: {self.weight} Kg
    Width: {self.width} cm
    Breadth: {self.breadth} cm
    Height: {self.height} cm
    Noice: {self.noice}
    Diet is: {self.diet}
    Is hairy: {self.is_hairy}
    Is trained: {self.is_trained}
    Is scent dog: {self.is_scent_dog}
        """

# Inherits Cat from Domestic animals.
class Cat(Domestic_animal):

    def __init__(self):
        Domestic_animal.__init__(self)
        self.is_agressive = True

    # Calling this function changes the boolean value.
    def set_agression(self):

        if self.is_agressive:
            self.is_agressive = False
        else:
            self.is_agressive = True

    #Defining Cat printing.
    def __str__(self):

        return f"""
    ID: {self.id}
    Species: {self.species}
    Name: {self.name}
    Size: {self.size} m^3
    Weight: {self.weight} Kg
    Width: {self.width} cm
    Breadth: {self.breadth} cm
    Height: {self.height} cm
    Noice: {self.noice}
    Diet is: {self.diet}
    Is_hairy: {self.is_hairy}
    Is trained: {self.is_trained}
    Is agressive: {self.is_agressive}
        """

```

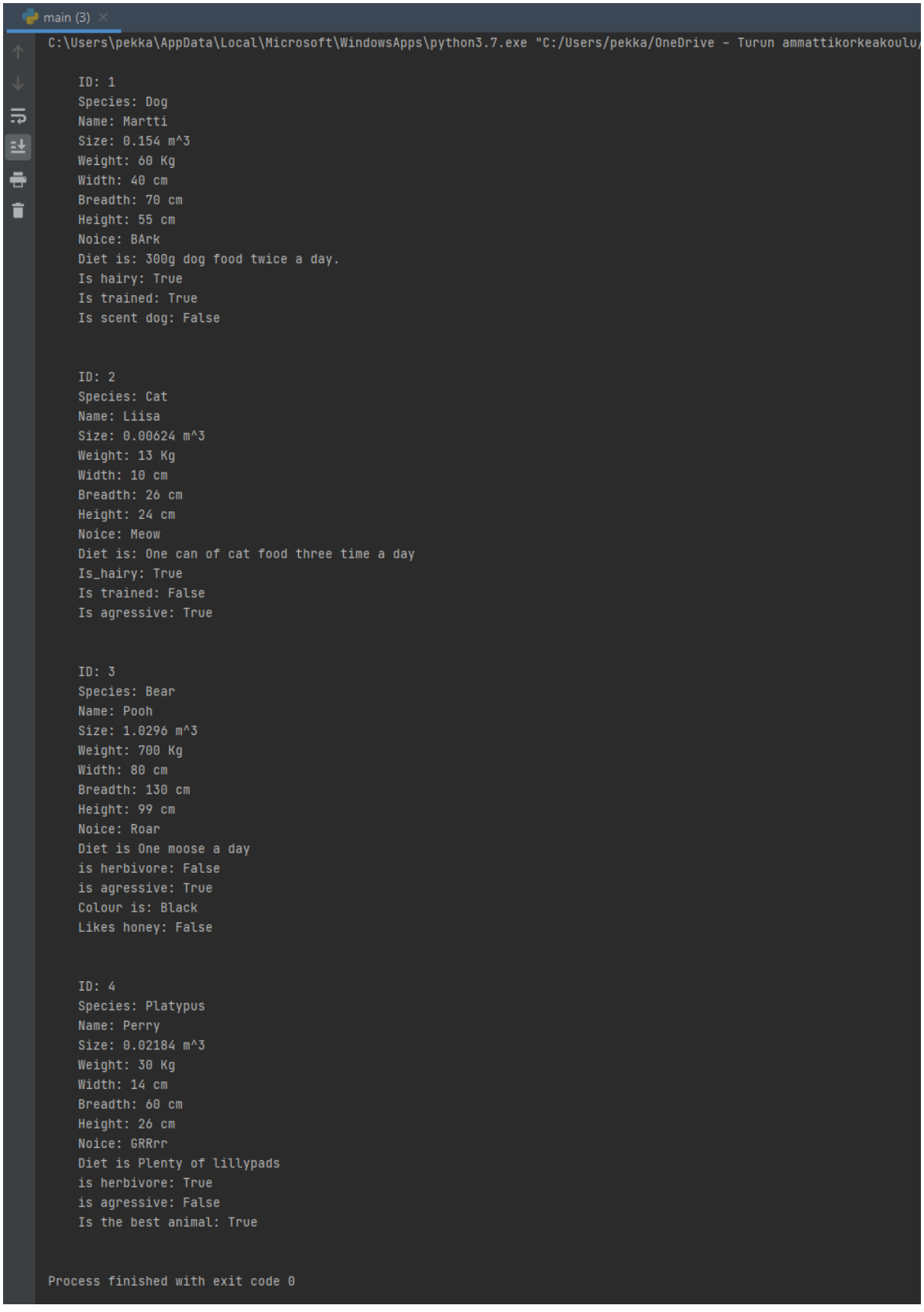
Main part 1:

```python
# File name: main
# Author: Pekka Lehtola
# Description: Main function for exercise 6_5

#Importing Domestic_animal and Wild_animal classes.
from domestic_animals import *
from wild_animals import *

#Cretes dog object with given attributes.
def dog_creating():

    dog = Dog()

    dog.set_id(1)
    dog.set_species("Dog")
    dog.set_name("Martti")
    dog.set_weight(60)
    dog.set_width(40)
    dog.set_breadth(70)
    dog.set_height(55)
    dog.set_noice("BArk")
    dog.set_diet("300g dog food twice a day.")
    dog.set_is_scent()

    print(dog)

#Cretes cat object with given attributes.
def cat_creating():

    cat = Cat()

    cat.set_id(2)
    cat.set_species("Cat")
    cat.set_name("Liisa")
    cat.set_weight(13)
    cat.set_width(10)
    cat.set_breadth(26)
    cat.set_height(24)
    cat.set_noice("Meow")
    cat.set_training()
    cat.set_diet("One can of cat food three time a day")

    print(cat)
```

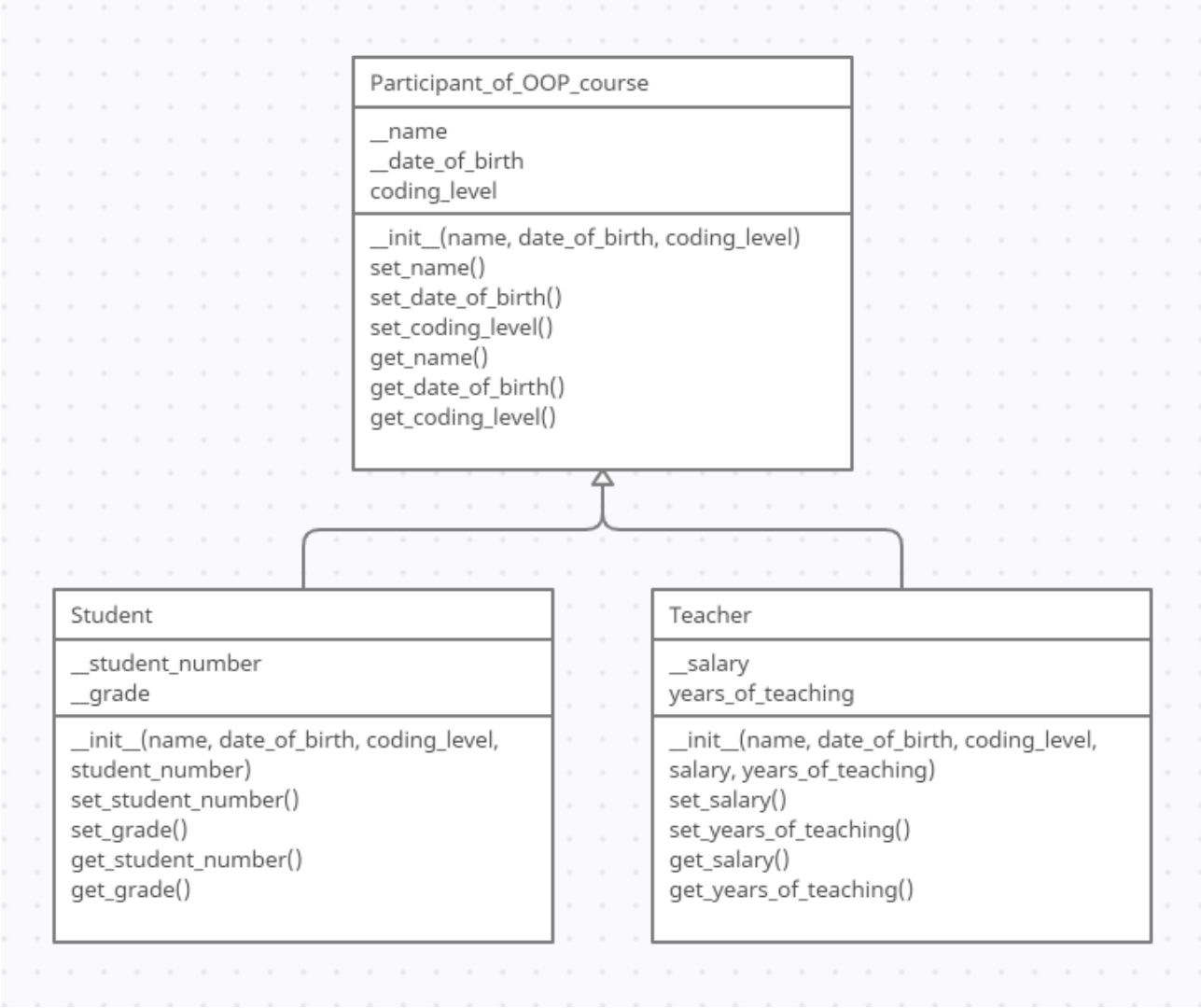
Main part 2:

```python
#Cretes bear object with given attributes.
def bear_creation():

    bear = Bear()

    bear.set_id(3)
    bear.set_species("Bear")
    bear.set_name("Pooh")
    bear.set_weight(700)
    bear.set_width(80)
    bear.set_breadth(130)
    bear.set_height(99)
    bear.set_noice("Roar")
    bear.set_diet("One moose a day")
    bear.set_herbivore()
    bear.set_agression()
    bear.set_colour("Black")
    bear.set_likes_honey()

    print(bear)

#Cretes platypus object with given attributes.
def platypus_creation():

    platypus = Platypus()

    platypus.set_id(4)
    platypus.set_species("Platypus")
    platypus.set_name("Perry")
    platypus.set_weight(30)
    platypus.set_width(14)
    platypus.set_breadth(60)
    platypus.set_height(26)
    platypus.set_noice("GRRrr")
    platypus.set_diet("Plenty of lillypads")


    print(platypus)


dog_creating()
cat_creating()
bear_creation()
platypus_creation()
```

Screen capture of the output of Task 5



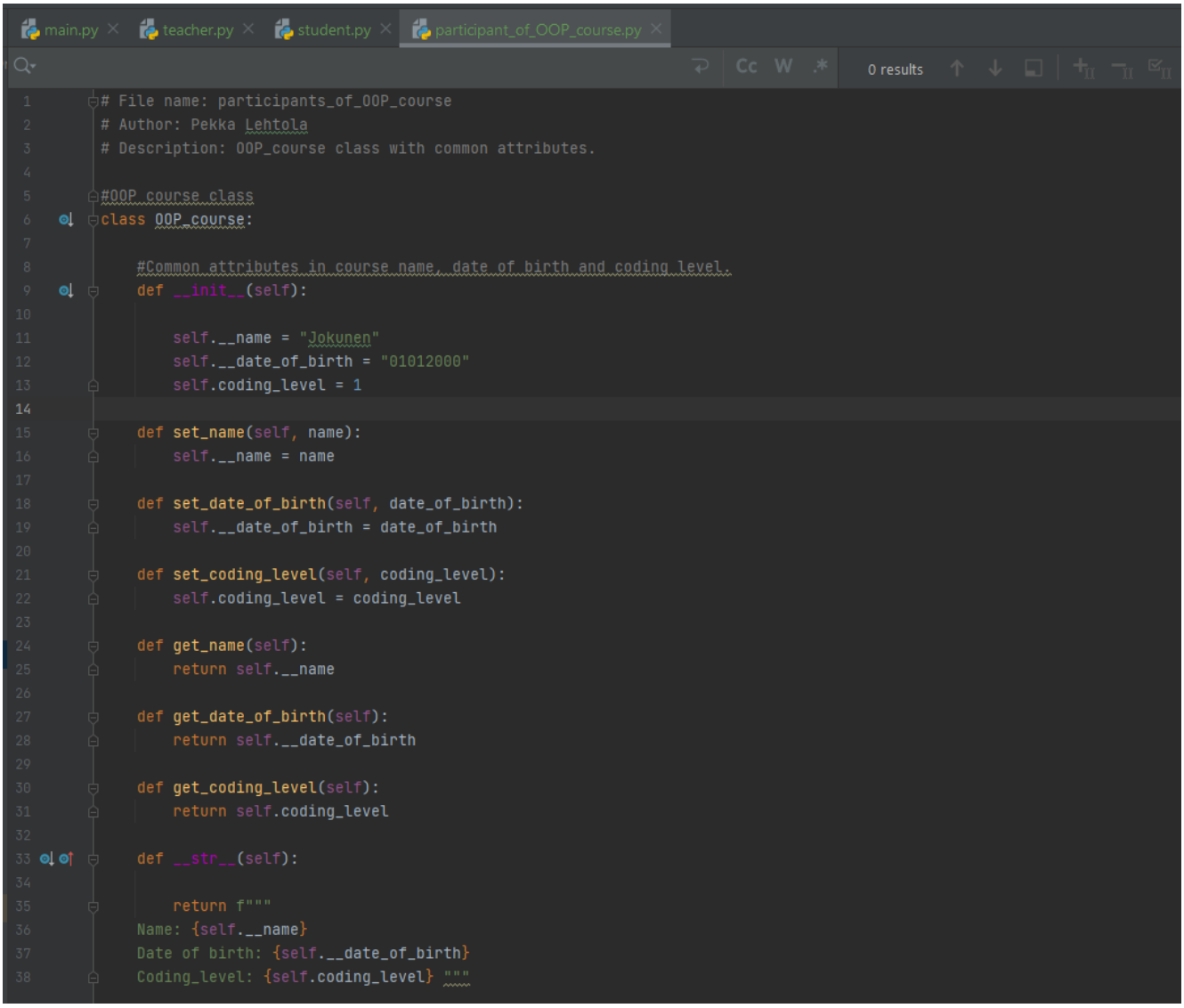
6. Draw a UML diagram of tasks 7. Use UML syntax and see how data attributes and methods are presented in the example presented on class (about Automobile etc.).



7. Inherit a student and teacher from a participant of OOP course. Think a few proper data attributes that are 1) common for both teachers and students and 2) different between teachers and students.

Screen capture of Task 7

Participant_of_OOP_course:



Student:

main.py | teacher.py | student.py | participant_of_OOP_course.py

0 results

```python
# File name: student
# Author: Pekka Lehtola
# Description: Student class inherited from OOP_course class

from participant_of_OOP_course import *

#Student class inherited from OOP course
class Student(OOP_course):
    #Student has additional attributes student number and grade
    def __init__(self):

        OOP_course.__init__(self)
        self.__student_number = "000000"
        self.__grade = "N/A"

    def set_student_number(self, student_numer):
        self.__student_number = student_numer

    def set_grade(self, grade):
        self.__grade = grade

    def get_student_number(self):
        return self.__student_number

    def get_grade(self):
        return self.__grade

    def __str__(self):
        return f"""

    Name: {self.get_name()}
    Date of birth: {self.get_date_of_birth()}
    Coding_level: {self.coding_level}
    Student number: {self.get_student_number()}
    Grade: {self.get_grade()} """
```

Teacher:

main.py | teacher.py | student.py | participant_of_OOP_course.py

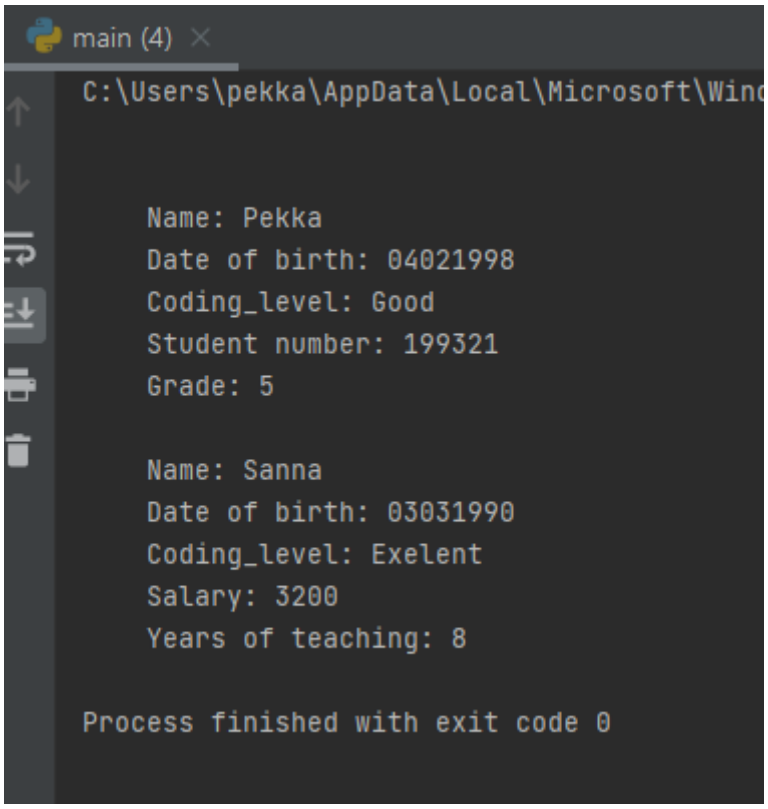
```python
# File name: teacher
# Author: Pekka Lehtola
# Description: Teacher class inherited from OOP_course class

from participant_of_OOP_course import *

#Teacher class inherited from OOP course
class Teacher(OOP_course):
    #Teacher has additional attributes salary and years of teaching.
    def __init__(self):
        OOP_course.__init__(self)
        self.__salary = 0
        self.years_of_teaching = 0

    def set_salary(self, salary):
        self.__salary = salary

    def set_years_of_teaching(self, years_of_teaching):
        self.years_of_teaching = years_of_teaching

    def get_salary(self):
        return self.__salary

    def get_years_of_teaching(self):
        return self.years_of_teaching

    def __str__(self):

        return f"""
    Name: {self.get_name()}
    Date of birth: {self.get_date_of_birth()}
    Coding_level: {self.coding_level}
    Salary: {self.get_salary()}
    Years of teaching: {self.years_of_teaching} """
```

Main:

main.py | teacher.py | student.py | participant_of_OOP_course.py

```python
# File name: main
# Author: Pekka Lehtola
# Description: Main file for exercise6_7

from student import *
from teacher import *

#Creates student object from Student class
def student_creation():

    pekka = Student()

    #Attribute values are set here.
    pekka.set_name("Pekka")
    pekka.set_date_of_birth("04021998")
    pekka.set_coding_level("Good")
    pekka.set_student_number("199321")
    pekka.set_grade(5)

    #Object printing
    print(pekka)

#Creates Teacher object from Teacher class
def teacher_creation():

    sanna = Teacher()

    sanna.set_name("Sanna")
    sanna.set_date_of_birth("03031990")
    sanna.set_coding_level("Exelent")
    sanna.set_salary(3200)
    sanna.set_years_of_teaching(8)

    print(sanna)

#Function activation happens here.
student_creation()
teacher_creation()
```

Screen capture of the output of Task 7



8. Each participant of Task 7 has also 1 domestic animal and 1 wild animal. Display the teachers, students and their information.

Files that are unchanged : animals, mammal_class, wild_animals, teacher, student, wild_animals, domestic_animals

Screen capture of Task 8

Small edits to animal creation functions **(Main in task 5)** and person creation functions **(Main in task 7):**

```
#Cretes dog object with given attributes.
def dog_creating():

    global dog

    dog = Dog()
#Cretes cat object with given attributes.
def cat_creating():

    global cat

    cat = Cat()
#Cretes bear object with given attributes.
def bear_creation():

    global bear

    bear = Bear()
#Cretes platypus object with given attributes.
def platypus_creation():

    global platypus

    platypus = Platypus()
```

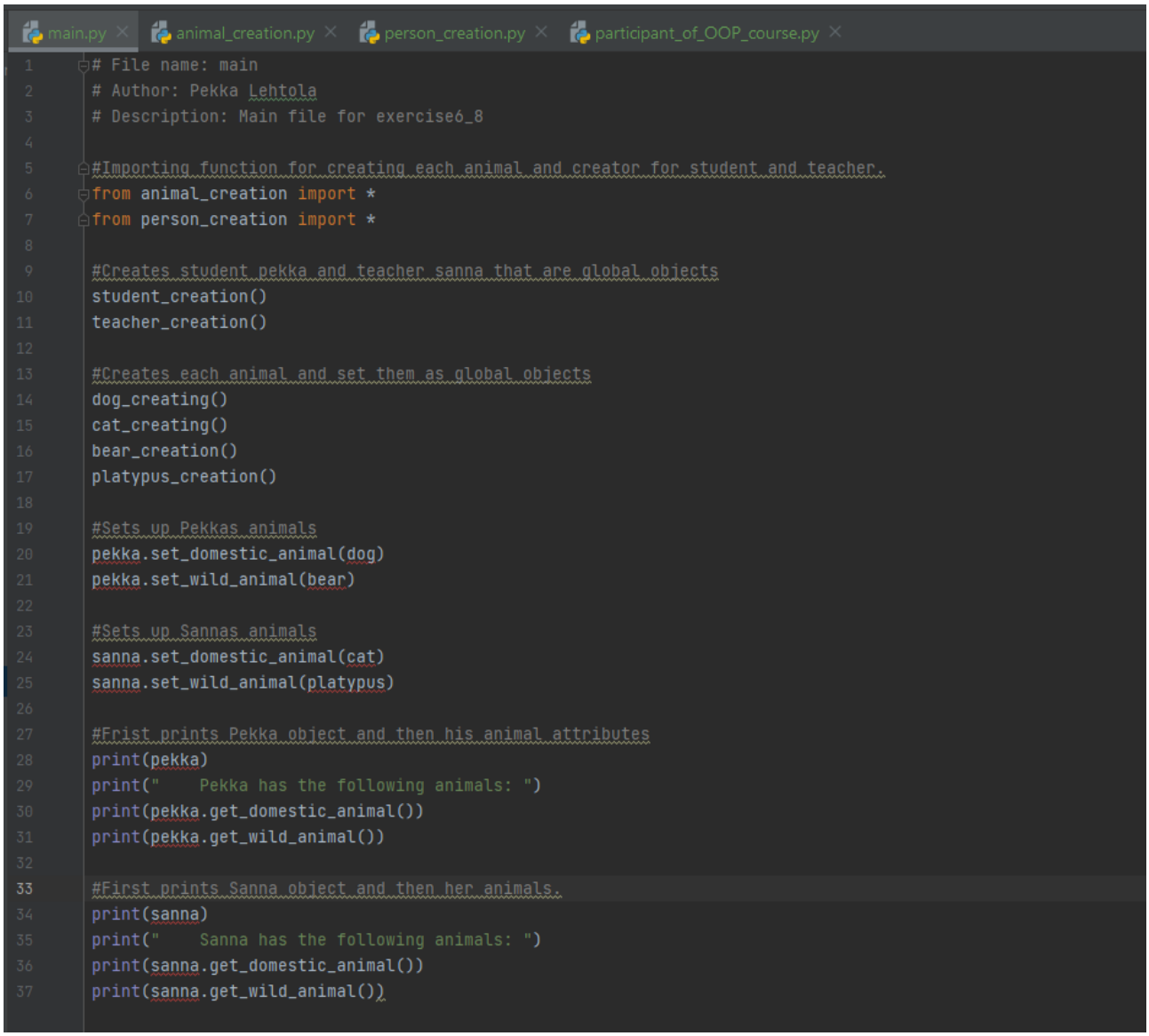
```
def student_creation():

    global pekka

    pekka = Student()
#Creates Teacher object from Teacher class
def teacher_creation():

    global sanna

    sanna = Teacher()
```

Edits to OOP_course class (From task 7):

```python
#OOP course class
class OOP_course:

    #Common attributes in course name, date of birth and coding level.
    def __init__(self):

        self.__name = "Jokunen"
        self.__date_of_birth = "01012000"
        self.coding_level = 1
        self.domestic_animal = None
        self.wild_animal = None
```

```python
36
37      def set_domestic_animal(self, domestic):
38          self.domestic_animal = domestic
39
40      def set_wild_animal(self, wild):
41          self.wild_animal = wild
42
43      def get_domestic_animal(self):
44          return self.domestic_animal
45
46      def get_wild_animal(self):
47          return self.wild_animal
48
```

Main:



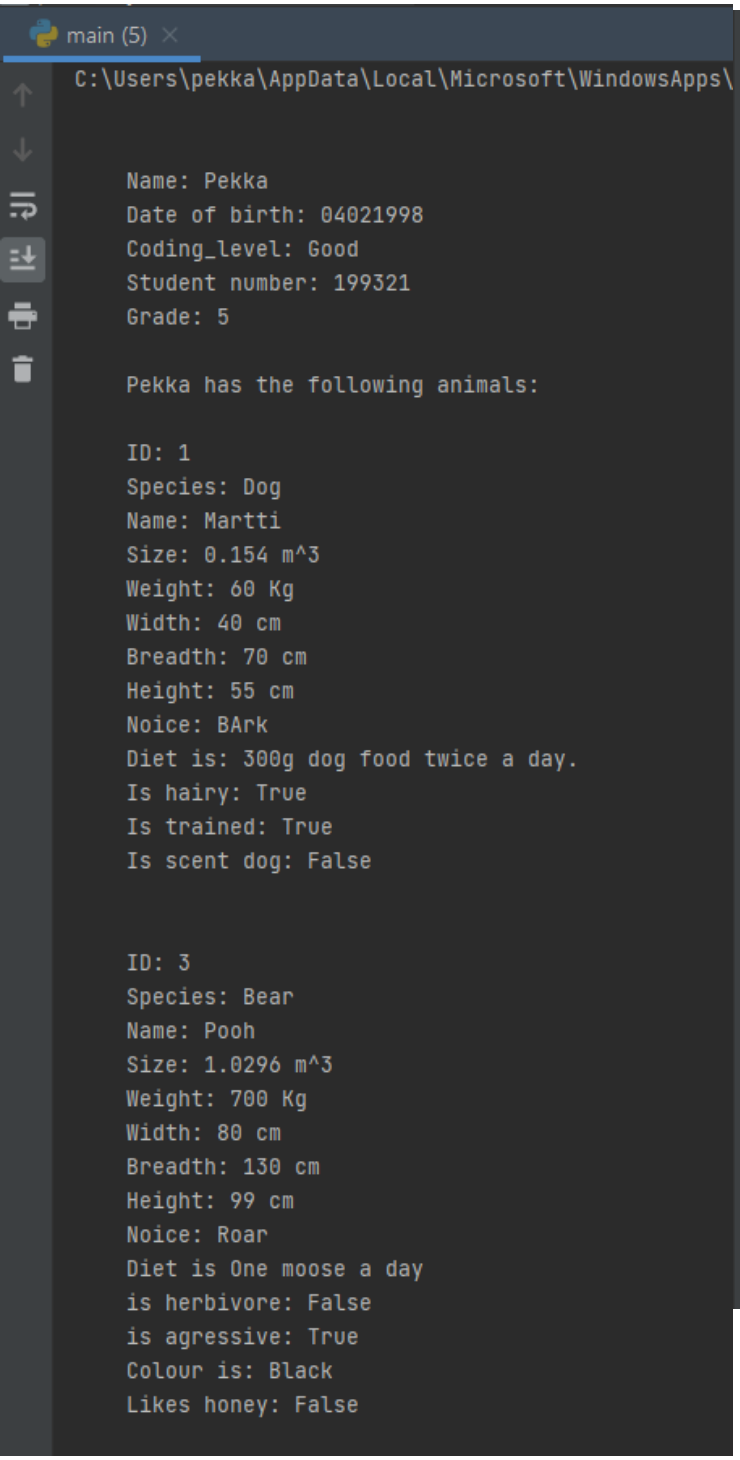

```python
# File name: main
# Author: Pekka Lehtola
# Description: Main file for exercise6_8

#Importing function for creating each animal and creator for student and teacher.
from animal_creation import *
from person_creation import *

#Creates student pekka and teacher sanna that are global objects
student_creation()
teacher_creation()

#Creates each animal and set them as global objects
dog_creating()
cat_creating()
bear_creation()
platypus_creation()

#Sets up Pekkas animals
pekka.set_domestic_animal(dog)
pekka.set_wild_animal(bear)

#Sets up Sannas animals
sanna.set_domestic_animal(cat)
sanna.set_wild_animal(platypus)

#Frist prints Pekka object and then his animal attributes
print(pekka)
print("    Pekka has the following animals: ")
print(pekka.get_domestic_animal())
print(pekka.get_wild_animal())

#First prints Sanna object and then her animals.
print(sanna)
print("    Sanna has the following animals: ")
print(sanna.get_domestic_animal())
print(sanna.get_wild_animal())
```

Error highlights are caused from computer not recognizing global objects.

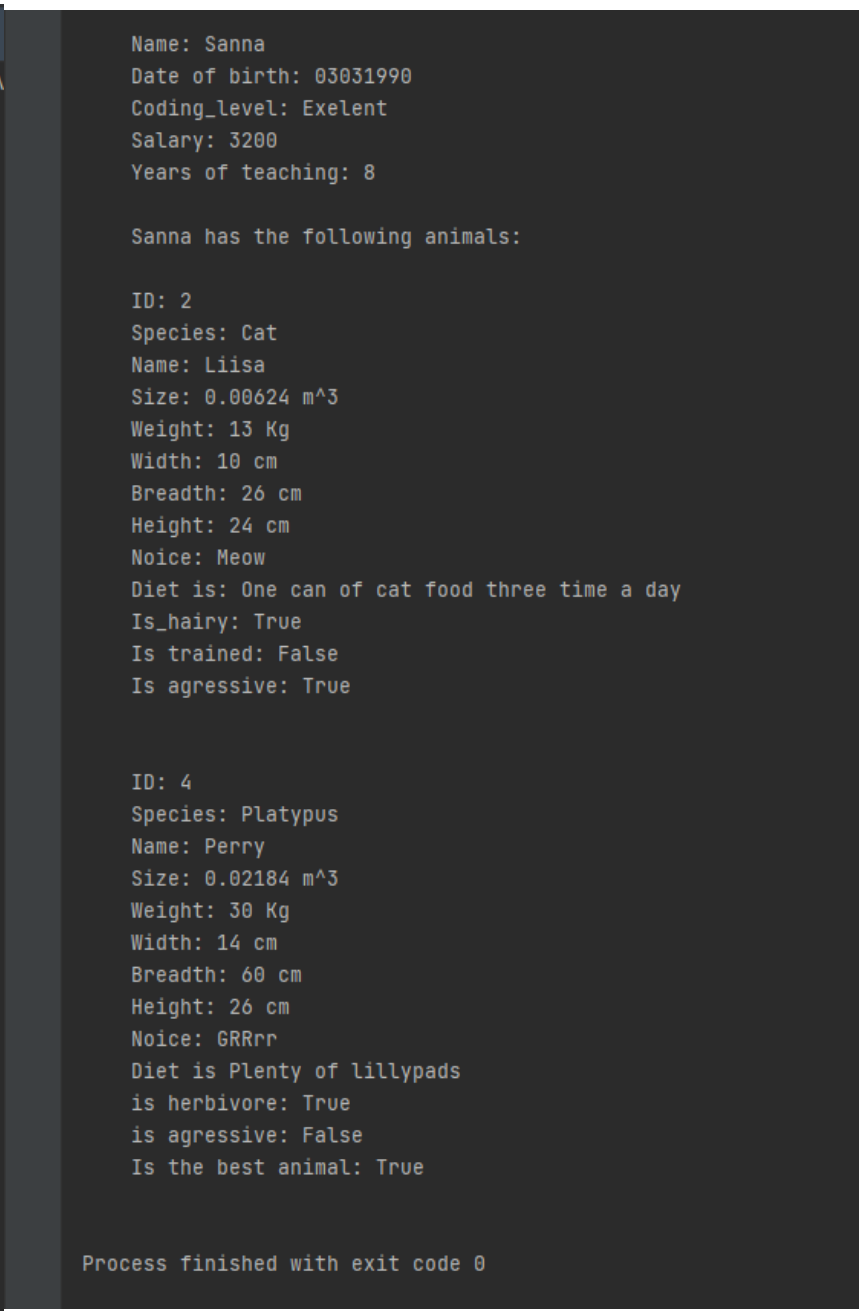
I didn’t add all the code here because document would have been way too long.

Screen capture of the output of Task 8

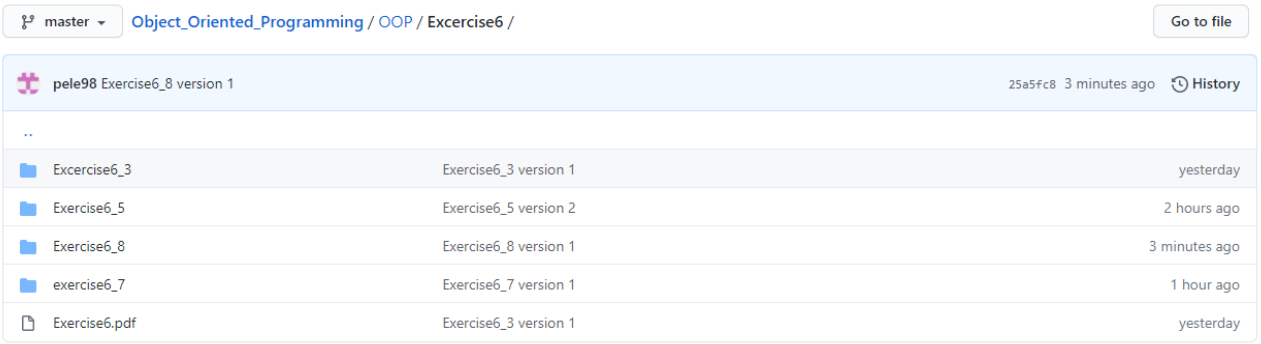
Output part 1:



Output part 2:



Screen capture of git log (showing that you made a commit after every task).



Self-assessment:

This exercise was easy/difficult/ok/etc. for me because…

Nämä tehtävä koin melko helpoiksi, haastetta kyllä toi se kun jaoin luokat ja funktiot eri tiedostoihin ja sen kautta tuli muutama ongelma.

Doing this exercise, I learned…

Periytyvyyttä ja sen että objectin attribuutti voi olla myös kokonainen objecti.

Oppisin myös käyttämään UML charttia jonka huomasin olevan paljon kätevämpi koodin hahmottelussa, kuin pseudo koodi.

I am still wondering…

Voiko __str__ function periä myös ja siihen tehdä muokkauksia perityn luokan sisällä.

I understood/did not understand that… ; I did/did not know that… ; I did/did not manage to do…

-